(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) N° de publication : **2 801 705**
(à n'utiliser que pour les commandes de reproduction)

(21) N° d'enregistrement national : **99 14882**

(51) Int Cl[7] : G 06 F 19/00

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 26.11.99.

(30) **Priorité :**

(43) **Date de mise à la disposition du public de la demande :** 01.06.01 Bulletin 01/22.

(56) **Liste des documents cités dans le rapport de recherche préliminaire :** *Se reporter à la fin du présent fascicule*

(60) **Références à d'autres documents nationaux apparentés :**

(71) **Demandeur(s) :** *BULL SA Société anonyme* — FR.

(72) **Inventeur(s) :** ANDREI PIERRE et BUI XUAN HOAN.

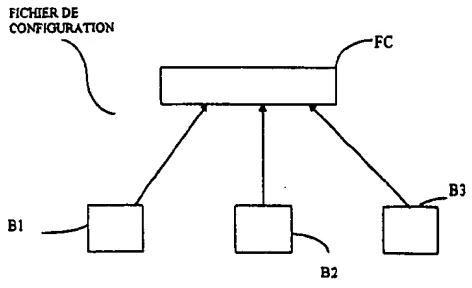
(73) **Titulaire(s) :**

(74) **Mandataire(s) :**

(54) PROCEDE DE CREATION DE FICHIERS DE CONFIGURATION D'OBJETS INCLUS DANS UN SYSTEME INFORMATIQUE.

(57) L'invention a pour objet la création d'au moins un fichier de configuration d'au moins un objet matériel et/ ou logiciel comprenant des paramètres, ledit fichier de configuration étant écrit en utilisant un métalangage de description dont le format est indépendant du matériel et/ ou logiciel à configurer, ce fichier de configuration incluant tout ou partie des paramètres dudit objet et se reposant sur un fichier de description définissant des contraintes à respecter sur sa structure et sa syntaxe lors de son écriture. L'invention consiste, avant l'écriture du fichier de configuration,

- à étendre le fichier de description par au moins un modèle comprenant au moins un paramètre décrit dans le fichier de description,
- et à valoriser tout ou partie des paramètres de ce modèle.



FR 2 801 705 - A1

Procédé de création de fichiers de configuration d'objets inclus dans un système informatique.

DESCRIPTION

Domaine technique

La présente invention se rapporte à un procédé de création de fichiers de configuration d'objets appartenant à un système informatique.

Le système informatique comprend des objets matériels (machines, ...), et/ou logiciels (applications, ...). Ce système est indifféremment un système distribué ou non, hétérogène ou non.

Les objets comprennent des paramètres inclus dans un fichier de configuration. L'invention s'applique à tout fichier de configuration écrit dans un langage de balisage extensible, c'est-à-dire un langage qui présente de l'information encadrée par des balises. Le fichier de configuration est écrit en utilisant un métalangage de description dont le format est indépendant du matériel et/ou logiciel à configurer. Un métalangage se défini généralement comme étant un langage utilisé pour décrire un autre langage. Le langage de balisage XML (eXtensible Markup Langage), connu de l'homme du métier, est bien adapté à la mise en œuvre de la présente solution. Rappelons que la spécification du langage XML est définie par le consortium W3C (World Wide Web Consortium). Ce consortium est un Organisme de promotion du «World Wide Web», qui met au point des normes et des protocoles ouverts et libres, dans un souci d'Interopérabilité maximale. Le fichier de configuration XML a une structure déclarée dans un fichier de description. Ce fichier de description comprend la description des paramètres de configuration d'un objet et se nomme généralement Définition de Type de Document (DTD). Cette définition

s'effectue selon un formalisme particulier également défini dans la spécification XML du consortium W3C.

L'art antérieur

Par définition, l'écriture d'un fichier de configuration dans un langage XML doit obéir à certaines contraintes syntaxiques. En effet, un document XML a une structure logique. Il se compose de descriptions, d'éléments, de commentaires, d'appels de caractère et d'instructions de traitement, qui sont indiqués dans le document par l'intermédiaire d'un balisage explicite. Les éléments sont encadrées par de balises ouvrantes, par exemple <préface>, et des balises fermantes, par exemple </préface>. Les éléments sont structurés et décrivent les paramètres des objets. Les paramètres, comme un attribut d'un objet, peuvent être inclus à l'intérieur même des balises. Par exemple, on peut écrire <LIVRE sujet = k> signifiant que l'attribut sujet de l'élément LIVRE a la valeur K.

Dans notre exemple de réalisation, Les paramètres des objets sont définis dans un fichier de configuration écrit dans un langage XML tel que défini précédemment.

Le problème principal est que les objets à configurer se comptent en milliers et que plusieurs de ces objets peuvent se reposer sur un même fichier de description DTD et avoir des valeurs identiques pour tout ou partie de leurs paramètres. Pour construire le fichier de configuration, l'administrateur du système de gestion doit alors valoriser les paramètres décrit dans le fichier de description autant de fois qu'il y a d'objets se reposant sur ce fichier DTD. Plus précisément, supposons que deux objets B1 et B2 se reposent sur un même fichier de description (DTD). Pour configurer l'objet B1, l'utilisateur doit valoriser tous les paramètres décrit dans le fichier DTD. Pour configurer l'objet B2, l'utilisateur doit à nouveau valoriser tous les paramètres décrit dans le fichier DTD. En conséquence, les paramètres de même valeur sont écrit autant de fois qu'il y a d'objets se reposant sur un même fichier de description.

L'écriture d'un fichier de configuration présente donc des redondances. Le langage XML étant un langage de configuration da bas niveau, un utilisateur est donc contraint d'écrire dans le fichier de configuration des descriptions de paramètres répétitives dont la sémantique est, dans certains cas, sans rapport avec ses besoins, ce qui nécessite une culture importante de sa part de l'ensemble des syntaxes offertes par le langage XML. De ce fait, le fournisseur doit fournir avec le fichier de description une documentation précise. Il est clair qu'un fichier de configuration impose un coût en temps important en écriture.

Un autre problème, lié au nombre important d'objets à décrire, est que si un utilisateur situé sur une machine quelconque du réseau souhaite visualiser des ressources du système informatique configurées dans le fichier de configuration, le système de gestion doit alors transmettre le fichier de configuration à la machine distante par l'intermédiaire du réseau. Lorsque le fichier de configuration a un gros volume, le flux de données entre le système de gestion et l'application cliente peut s'avérer très important et conduire à saturer le système de communication entre les applications clientes et le système de gestion.

Enfin, un autre problème est que la syntaxe définie selon les recommandations du consortium W3C doit être respectée à la lettre tout au long de l'écriture du fichier de configuration. Le risque d'erreur lors de l'écriture d'un fichier de configuration est donc permanent pour un administrateur du système de gestion.

Sommaire de l'invention

Un premier but de la solution est donc de simplifier considérablement l'écriture des fichiers de configuration réduisant en conséquence à la fois le coût en temps d'écriture de celui-ci et le risque d'erreurs d'écriture.

Un deuxième but visé est de réduire la taille du fichier de configuration.

A cet effet, la solution a pour objet un procédé de création d'au moins un fichier de configuration d'objets matériels et/ou logiciels présents dans un

système informatique, ledit fichier de configuration étant écrit en utilisant un métalangage de description dont le format est indépendant du matériel et/ou logiciel à configurer, ce fichier de configuration incluant tout ou partie des paramètres desdits objets et se reposant sur un fichier de description définissant des contraintes à respecter sur la structure et la syntaxe lors de l'écriture dudit fichier de configuration, caractérisé en ce qu'il consiste à étendre le fichier de description par au moins un modèle comprenant au moins un paramètre décrit dans le fichier de description, et en ce qu'il consiste à valoriser tout ou partie des paramètres de ce modèle.

Il en résulte également un fichier de configuration d'objets matériels et/ou logiciels présents dans un système informatique, ledit fichier de configuration étant écrit en utilisant un métalangage de description dont le format est indépendant du matériel et/ou logiciel à configurer, ce fichier de configuration incluant tout ou partie des paramètres desdits objets et se reposant sur un fichier de description définissant des contraintes à respecter sur la structure et la syntaxe lors de l'écriture dudit fichier de configuration, caractérisé en ce que le fichier de description est étendu, en ce que l'extension comprend au moins un modèle incluant au moins un paramètre inclus dans le fichier de description, et en ce que une partie des paramètres de ce modèle sont valorisés.

L'invention sera mieux comprise à la lecture de la description qui suit, donnée à titre d'exemple et faite en référence aux dessins annexés.

Description d'un exemple de réalisation

Dans les dessins:

- la figure 1 est une vue synoptique de l'architecture d'un système informatique sur lequel peut s'appliquer la solution,

- la figure 2, est une vue d'un modèle conforme à la présente solution.

Sur la figure 1, on a représenté un système informatique SYS distribué illustrant un exemple de réalisation préféré de la solution. Dans l'exemple illustré, ce système SYS inclut un système de gestion SG et au moins une machine M1. Le système de gestion SG comprend au moins un système d'exploitation, au moins une mémoire de stockage d'informations et au moins un processeur contrôlant le processus du traitement de l'information. Le terme gestion est utilisé pour être conforme à la traduction Afnor (Association Française de NORmalisation) de « Management ». Un système de gestion de machines de type « Open Master » (marque déposée par la société BULL S.A.), connu de l'homme du métier, est particulièrement bien adapté pour la mise en œuvre de la solution. Ce système de gestion peut être assimilé à un ensemble de services qui interagissent entre eux pour donner une représentation objet du monde réel constitué notamment par les machines du système informatique. C'est une représentation objet qu'un administrateur manipule (surveillance, action) pour gérer le monde réel. La représentation objet porte sur des objets virtuels du monde réel et constitue un modèle objet. En d'autres mots, un objet géré par le système de gestion est une vue abstraite, définie pour les besoins de gestion, d'une ressource logique ou physique du système informatique. (disque, processeur, mémoire, etc.) et/ou logiques (fichiers, processus, sémaphores, etc.).

Le système de gestion et les machines qu'il gère constitue une architecture Client/Serveur. Dans une telle architecture, un application cliente interroge le système de gestion pour connaître l'état des objets gérés par le système de gestion. Le mode client/Serveur a l'avantage de permettre à un utilisateur appelé client (ou application cliente) situé sur une machine, par exemple par l'intermédiaire d'un simple micro-ordinateur ou d'une station de travail, de confier une partie de sa tâche ou de ses opérations à effectuer au serveur à savoir le système de gestion. De cette manière, le client dispose d'une capacité de calcul beaucoup plus importante que celle de son micro-ordinateur.

Le système informatique peut être hétérogène. Afin de masquer l'hétérogénéité du système informatique, le système de gestion SG et les machines gérées par le système de gestion comprennent au moins un agent respectif associé à un protocole de gestion. Un agent assure, entre autres, une conversion de protocole.

Le système de gestion est relié à une machine gérée par l'intermédiaire d'un réseau quelconque. Le réseau peut être de type LAN (Local Area Network), WAN (Wide Area Network). Un ensemble de couches logicielles s'interpose entre le système de gestion SG et le réseau RES et entre le réseau et chaque machine. Pour des raisons de simplification de la description, cet ensemble de couches logicielles n'est pas représenté sur la figure 1.

Chaque objet géré comprend des paramètres définis dans un fichier de configuration de préférence écrit dans un langage de description à balisage comportant une structure et incluant tout ou partie des paramètres (nom, au moins un attribut, au moins une action, etc.) des objets. Le fichier de configuration se base sur un fichier distinct appelé fichier de description définissant les contraintes sur la structure et les contraintes syntaxiques des paramètres pour l'écriture dudit fichier de configuration associé à un objet d'une machine. De préférence, le fichier de configuration est écrit dans un langage de type XML, et le fichier de description est un fichier de description de type DTD, connus de l'homme du métier. Dans notre exemple de réalisation, ce fichier de configuration XML et le fichier de description DTD sont inclus dans le système de gestion SG.

Dans notre exemple de réalisation, le fichier de description DTD est centralisé sur le système de gestion de façon à être utilisable par toutes les machines du réseau. De préférence, les objets gérés peuvent être représentés par un arbre, chaque noeud de l'arbre représentant un objet géré. Une application de visualisation APV incluse sur la machine M1 peut interroger le fichier de configuration dans le but de recevoir les paramètres des objets configurés et de visualiser ces objets sur cette machine. De préférence, pour la visualisation des paramètres de configuration des objets, la machine M1 est

munie d'un navigateur standard dit « Internet Explorer », connu de l'homme du métier, dans lequel un programme en langage JAVA est exécuté pour lire le fichier de configuration, accédant ainsi à son contenu et à sa structure, et pour transmettre les informations lues à (aux) applications de visualisation APV.

Description de paramètres de configuration d'un objet :

Un objet comprend comme paramètre au moins un attribut. Par exemple, un attribut ID est l'identificateur de l'objet, un autre attribut TYPE désigne son type, et un autre attribut OWNER. De plus, un objet a des propriétés. Dans notre exemple, un objet comprend également comme paramètres:

- le nom de l'objet,
- les actions que l'on peut exécuter sur cet objet incluant l'action ouvrir pour l'ouverture du noeud, l'action fermer pour la fermeture du noeud, l'action développer pour visualiser les noeuds subordonnés à un noeud,
- et des propriétés graphiques de ce noeud incluant le type de police de caractère, l'adresse de l'icône qui lui est associé, et la couleur de fond désirée

Ainsi, dans ce fichier de description DTD, un élément « noeud » est associé à un noeud, et des éléments lui sont subordonnés et sont associés respectivement

- au nom de l'objet
- aux actions (ouvrir, fermer, développer) que l'on peut exécuter sur cet objet,
- et aux propriétés graphiques (police de caractère, icône, couleur de fond) de cet objet.

Des attributs peuvent être associé à un élément. Dans notre exemple de réalisation, l'élément « noeud » comprend trois attributs à savoir :

- un attribut ID désignant son identificateur
- un attribut TYPE désignant son type
- et un attribut désignant son propriétaire

Un fichier de description DTD définissant un objet de l'arbre peut être écrit de la façon suivante, en respectant le formalisme particulier défini dans la spécification XML du consortium W3C :

```
< !ELEMENT noeud (noeudNom, noeudActions, noeudPropriétéGraphique)>
< !ELEMENT noeudActions (action*)>
< !ELEMENT noeudNom (groupeNom, adresseNom, versionNom)>
< !ATTLIST noeud Id CDATA #REQUIRED
                    Type CDATA #REQUIRED
                    Owner CDATA #REQUIRED
>
< !ELEMENT noeudPropriétégraphique (police, icône, couleur)>
```

Dans ce fichier, CDATA #REQUIRED siginifie que l'attribut en question doit être un bloc de texte contenant des caractères. De plus, l'écriture « action* » signifie que les actions n'ont pas d'attributs et que la syntaxe de ces actions à utiliser lors de l'écriture du fichier de configuration est du texte.

L'écriture < !ELEMENT noeudNom (groupeNom, adresseNom, versionNom)> indique que l'élément « noeudNom » comprend trois éléments (groupeNom, adresseNom, versionNom) qui lui sont subordonné. L'élément « groupeNom » désigne le nom de l'objet, l'élément « adresseNom » désigne l'adresse de l'objet, et l'élément « versionNom » désigne la version de l'objet.

Ce fichier de description correspondra dans la suite de la description au fichier de description initialement créé.

Notons que le nombre de paramètres dans notre exemple de réalisation est réduit pour des raisons de simplification de la description. En général un fichier de description comprend un nombre de paramètres plus important.

Dans l'exemple illustré, supposons par exemple que trois objets (OBJ1, OBJ2 et OBJ3) se reposent sur le même fichier de description DTD défini plus

haut. Le problème principal est que, pour construire le fichier de configuration, l'administrateur du système de gestion doit valoriser les paramètres décrit dans le fichier de description autant de fois qu'il y a d'objets se reposant sur ce fichier DTD. L'administrateur doit donc compléter le fichier de configuration trois fois. Le coût en temps d'une telle écriture est donc considérable.

A cet effet, la solution consiste à étendre le fichier de description par au moins un modèle comprenant au moins un paramètre inclus dans le fichier de description, et en ce qu'il consiste à valoriser une partie des paramètres de ce modèle d'élément. En L'espèce, la solution consiste à introduire un modèle d'élément dont l'écriture respectent des propriétés définies dans ce qui suit.

Dans notre exemple de réalisation, nous avons à définir un ensemble d'objets dont les seuls paramètres variant entre eux sont

- l'élément « Nom » correspondant au nom des objets (OBJ1, OBJ2 et OBJ3), respectivement (JAZZ, POP, SOL),

- et l'attribut « ID » correspondant à l'identificateur des objets (OBJ1, OBJ2 et OBJ3), respectivement (123, 142, 162).

Ces deux paramètres seront dits indéfinis dans le suite de la description. Dans l'exemple de réalisation, les objets (OBJ1, OBJ2 et OBJ3) ont un nom respectif (JAZZ, POP, SOL) et un identificateur respectif (123, 142, 162).

Conformément à notre hypothèse de départ, les autres paramètres ont la même valeur pour chaque objet (OBJ1, OBJ2 et OBJ3). Ils seront dits définis. Ainsi,

- la valeur de l'élément correspondant aux actions que l'on peut exécuter est la même pour chaque objet (OBJ1, OBJ2 et OBJ3),
- la valeur de l'élément correspondant aux propriétés graphiques est la même pour chaque objet (OBJ1, OBJ2 et OBJ3) ,

et la valeur des attributs

■ TYPE désignant le type du objet est le même pour chaque objet (OBJ1, OBJ2 et OBJ3),

■ et OWNER désignant son propriétaire est le même pour chaque objet (OBJ1, OBJ2 et OBJ3).

Pour des raisons de simplification de la description, on donnera des valeurs arbitraires aux paramètres définis. Dans notre exemple, la valeur de l'élément correspondant aux actions que l'on peut exécuter sur chaque objet (OBJ1, OBJ2 et OBJ3) est ACT1 pour la commande « ouvrir », ACT2 pour la commande « fermer », et ACT3 pour la commande « développer ». De même, la valeur de l'élément correspondant aux propriétés graphiques de chaque objet (OBJ1, OBJ2 et OBJ3) est PRO1 pour la police de caractère, PRO2 pour l'icône, et PRO3 pour la couleur de fond. Enfin, les attributs TYPE et OWNER prennent pour chaque objet (OBJ1, OBJ2 et OBJ3) respectivement les valeurs « snmp » et « opérateur ».

Les deux étapes principales du procédé conforme à la solution sont respectivement

- l'écriture du fichier de description DTD et de l'extension associé à ce fichier constitué par au moins un modèle d'élément (étape 1),

- et l'écriture du fichier de configuration résultant de la valorisation des paramètres indéfinis du modèle d'élément (étape 2).

Ecriture du fichier de description et introduction de la notion de modèle d'élément dans un fichier de description :

La première étape doit réalisée en respectant certaines propriétés. Les paramètres de ces objets dont la valeur est invariante étant identifiés, la solution consiste à introduire le modèle d'élément MODELE dans le fichier de description DTD créé. Le modèle d'élément se distingue des autres éléments du fichier de description en ce sens qu'il comporte au moins un paramètre avec une valeur.

Premièrement, le modèle comprend, en respectant le formalisme d'écriture d'un fichier de description DTD défini dans la spécification XML du consortium W3C,

- une entête MODELE avec un nom spécifique « tnoeud »

- et une référence à un élément « noeud » défini du fichier de description DTD créé initialement.

Le modèle MODELE peut être écrit de la façon suivante :

< !MODELE tnoeud ELEMENT =noeud >

signifiant que le modèle MODELE à un nom spécifique « tnoeud » et qu'il se base sur la description de l'élément « noeud » défini au préalable dans le fichier de desciption (DTD).

Deuxièmement, dans ce modèle, l'écriture des éléments indéfinis est particulière. De façon à distinguer un élément défini d'un élément indéfini dans le modèle d'élément, les éléments à définir sont repérés dans ce modèle par l'intermédiaire d'une balise spécifique dont l'entête est par exemple < !DEFINIR...>. De plus, les éléments à définir sont identifiés par un nom « tnoeudNom » et une référence à un élément noeudNom du fichier de description initial précisant sur quel élément du fichier de description préalablement défini se base le modèle d'élément « tnoeudNom ». Dans notre exemple, un élément à définir peut s'écrire de la façon suivante :

< !DEFINIR tnoeudNom...ELEMENT noeudNom >.

A la différence des éléments indéfinis, les éléments définis du modèle d'élément sont écrits de la même façon que pour l'écriture d'un fichier de configuration XML.

Troisièmement, l'écriture des attributs indéfinis respecte des propriétés. Dans ce modèle, les attributs à définir et définis, la solution consiste à introduire deux mots clés DEFINIR et DEFINI indiquant qu'un paramètre

attribut est à définir (DEFINIR) ou est défini (DEFINI). Dans notre exemple, l'attribut ID est à définir lors de l'écriture du fichier de configuration, tandis que les attributs TYPE et OWNER sont définis et prennent respectivement les valeurs « snmp » et « opérateur ». Dans notre exemple, la liste d'attributs est relative au modèle d'élément MODELE « tnoeud » et peut s'écrire de la façon suivante :

```
< ! ATTLIST tnoeud
        ID DEFINIR
        TYPE DEFINI « snmp »
        OWNER DEFINI « opérateur »
>
```

En définitive, le fichier de description DTD qui découle d'une telle configuration peut s'écrire de la façon suivante :

```
< !MODELE  tnoeud ELEMENT=noeud
< !DEFINIR tnoeudNom ELEMENT=noeudNom>
< noeudActions >
        <action  nom=ouvrir> ACT1</action>
        <action  nom=fermer>ACT2</action>
        <action  nom=développer>ACT3</action>
</ noeudActions >
< noeudPropriétésgraphiques >
        <police de caractères PRO1... />
        <Icône>PRO2</Icône>
        < couleur de fond PRO3/>
< / Propriétésgraphiques >
>
< ! ATTLIST tnoeud
        ID DEFINIR
        TYPE DEFINI « snmp »
```

OWNER DEFINI « opérateur »

>.

Ce fichier constitue un facteur commun FC pour l'écriture du fichier de configuration et décrit les paramètres (élément et/ou attribut) à définir.

Ecriture du fichier de configuration

La figure 2 est une vue du fichier de configuration qui résulte de l'utilisation du modèle d'élément.

L'écriture du fichier de configuration correspond à la deuxième étape. Cette opération consiste à utiliser le modèle d'élément MODELE défini dans le fichier de description DTD et à valoriser les éléments et attributs indéfinis. Lors de l'écriture du fichier de configuration, la solution consiste à utiliser la partie comprenant les paramètres valorisés comme facteur commun, et en ce que l'écriture se limite à la valorisation des paramètres ne comportant pas de valeur.

En l'espèce, trois objets (OBJ1, OBJ2 et OBJ3) ayant pour nom respectif (JAZZ, POP, SOL) et un identificateur respectif (123, 142, 162) sont à configurer et l'écriture du fichier de configuration pour ces trois objets se limite à écrire les trois blocs suivants (B1, B2 et B3) :

(B1)

```
<tnoeud Id= «123»>
 < tnoeudNom >
    <noeudNom>
       <groupeNom> JAZZ </groupeNom>
      <adresseNom> db0 </ adresseNom >
       <versionNom> 8.0 <versionNom>
  </noeudNom>
```

```
    </ tnoeudNom >
  </tnoeud>
```

(B2)

```
  <tnoeud Id= «142»>
    < tnoeudNom >
      <noeudNom>
        <groupeNom> POP </groupeNom>
        <adresseNom> db1 </ adresseNom >
        <versionNom> 8.1 <versionNom>
      </noeudNom>
    </ tnoeudNom >
  </tnoeud>
```

(B3)

```
  <tnoeud Id= «162»>
    < tnoeudNom >
      <noeudNom>
        <groupeNom> SOL </groupeNom>
        <adresseNom> db2 </ adresseNom >
        <versionNom> 8.2 <versionNom>
      </noeudNom>
    </ tnoeudNom >
  </tnoeud>
```

Le nombre d'invocation du modèle d'élément correspond au nombre d'objets se reposant sur ce modèle MODELE. Ces trois blocs ont pour même facteur commun celui créé dans le fichier de description.

Selon une variante, un modèle d'élément peut contenir d'autres modèles d'éléments. Ainsi, dans un fichier de configuration, on peut utiliser à l'intérieur

d'une référence de modèle d'élément (par exemple « tnoeud ») une autre référence de modèle d'éléments. En effet, supposons qu'il existe dans le fichier de description DTD un modèle d'élément « tOracleNoeudNom » dont les éléments définis sont

- l'élément nommé groupeNom
- et l'élément versionNom

et dont l'élément indéfini est adresseNom. L'écriture de ce modèle d'élément peut être le suivant :

```
< !MODELE tOracle8NodeName ELEMENT=noeudNom
        <groupeNom> Oracle </groupeNom>
      < !DEFINIR tOracleDb ELEMENT=adresseNom>
        <versionNom> 8.0 <versionNom>
>
```

De cette façon, lors de l'écriture du fichier de configuration, on peut utiliser le modèle « tOracleNoeudNom » pour définir l'élément noeudNom. Le fichier de configuration s'écrit alors de la façon suivante :

```
<tnoeud Id= «123»>
  < tnoeudNom >
      < tOracle8NoeudNom>
          < tOracleDb>
          <adresseNom> db0 </adresseNom>
          </ tOracleDb>
      </ tOracle8NoeudNom>
  </ tnoeudNom >
</ tnoeud>
```

D'une manière générale, la solution a pour objet un procédé de création, dans un système informatique, d'au moins un fichier de configuration d'au moins un objet matériel et/ou logiciel comprenant des paramètres, ledit fichier

de configuration étant écrit en utilisant un métalangage de description dont le format est indépendant du matériel et/ou logiciel à configurer, ce fichier de configuration incluant tout ou partie des paramètres dudit objet et se reposant sur un fichier de description définissant des contraintes à respecter sur sa structure et sa syntaxe lors de son écriture, caractérisé en ce qu'il consiste, avant l'écriture du fichier de configuration, à étendre le fichier de description par au moins un modèle comprenant au moins un paramètre décrit dans le fichier de description, et à valoriser tout ou partie des paramètres de ce modèle.

Ensuite, lors de l'écriture du fichier de configuration, la solution consiste à utiliser la partie du modèle comprenant les paramètres valorisés comme facteur commun, et en ce que l'écriture du fichier de configuration se limite à la valorisation des paramètres ne comportant pas de valeur.

Lors de la création du modèle, on a vu que la solution peut consister tout d'abord à regrouper les objets se reposant sur le même fichier de description, ensuite à identifier les paramètres dont la valeur est identique entre tous ces objets, et enfin e à valoriser ces paramètres pour constituer un facteur commun dans ce modèle.

La solution consiste par exemple, lors de l'écriture du fichier de configuration, si au moins deux objets se reposent sur le même modèle, à utiliser le facteur commun et à valoriser uniquement le restant des paramètres de ce modèle autant de fois qu'il y a d'objets se reposant sur ce modèle d'élément.

Le langage utilisé est extensible. Dans notre exemple, on a vu que la solution consiste à donner un nom pour identifier le modèle dans le fichier de description, et en ce qu'il consiste à inclure dans le modèle une référence du fichier de description, cette référence définissant les contraintes à respecter sur la structure et la syntaxe de ce modèle. On a vu également que dans notre exemple, la solution consiste à introduire dans un modèle deux mots clés

DEFINIR et DEFINI indiquant qu'un paramètre est à définir (DEFINIR) ou est défini (DEFINI) dans ce modèle.

De préférence, le langage du fichier de configuration est le langage XML, la solution consistant à prendre comme paramètre un élément et/ou un attribut d'un objet. Selon cette variante, la solution consiste à étendre le fichier de description par au moins un modèle d'élément comprenant au moins un paramètre (élément et/ou attribut) décrit dans le fichier de description, et à valoriser tout ou partie des paramètres de ce modèle d'élément. De même, la solution consiste à donner un nom pour identifier le modèle d'élément dans le fichier de description, et en ce qu'il consiste à inclure dans le modèle une référence à un élément du fichier de description, cette référence définissant les contraintes à respecter sur la structure et la syntaxe de ce modèle.

Enfin, on a vu que, sur requête d'une application utilisant le fichier de configuration, la solution peut consister à transmettre le facteur commun et les blocs résultant de la valorisation des éléments indéfinis.

Enfin, il en résulte un fichier de configuration d'au moins un objet matériel et/ou logiciel comprenant des paramètres, ledit fichier de configuration étant écrit en utilisant un métalangage de description dont le format est indépendant du matériel et/ou logiciel à configurer, ce fichier de configuration incluant tout ou partie des paramètres dudit objet et se reposant sur un fichier de description définissant des contraintes à respecter sur sa structure et sa syntaxe lors de son écriture, caractérisé en ce que le fichier de description est étendu, et en ce que l'extension comprend au moins un modèle.

En conclusion, la solution offre de nombreux avantages. Un premier avantage est la simplification considérable de l'écriture d'un fichier de configuration. En effet, les paramètres définis étant valorisés dans le modèle d'élément, l'écriture du fichier de configuration se limite à la valorisation des paramètres indéfinis. En conséquence, l'administrateur évite ainsi la répétition de paramètres identiques entre objets. Le coût en temps d'écriture et le risque d'erreurs d'écriture lors de l'écriture du fichier de configuration est largement réduit. De plus, l'introduction d'un facteur commun dans le modèle réduit

considérablement la taille du fichier de configuration. Avantageusement, sur requête d'une application cliente, le système de gestion transmet le fichier de configuration incluant le facteur commun du modèle d'élément et, pour chacun des objets, les paramètres de ce modèle d'élément qui ont été valorisés lors de l'écriture du fichier de configuration. La taille du fichier de configuration étant réduit, le transfert de ce fichier s'effectue donc plus rapidement.

REVENDICATIONS

1- Procédé de création, dans un système informatique, d'au moins un fichier de configuration d'au moins un objet matériel et/ou logiciel comprenant des paramètres, ledit fichier de configuration étant stocké dans une mémoire de stockage d'information et étant écrit en utilisant un métalangage de description dont le format est indépendant du matériel et/ou logiciel à configurer, ce fichier de configuration incluant tout ou partie des paramètres dudit objet et se reposant sur un fichier de description définissant des contraintes à respecter sur sa structure et sa syntaxe lors de son écriture, caractérisé en ce qu'il consiste, avant l'écriture du fichier de configuration,.

- à étendre le fichier de description par au moins un modèle comprenant au moins un paramètre décrit dans le fichier de description,

- et à valoriser tout ou partie des paramètres de ce modèle.

2- Procédé selon la revendication 1, caractérisé en ce qu'il consiste, lors de l'écriture du fichier de configuration, à utiliser la partie du modèle comprenant les paramètres valorisés comme facteur commun, et en ce que l'écriture du fichier de configuration se limite à la valorisation des paramètres ne comportant pas de valeur.

3- Procédé selon la revendication 1 ou 2, caractérisé en ce qu'il consiste, lors de la création du modèle, à regrouper les objets se reposant sur le même fichier de description et ensuite à identifier les paramètres dont la valeur est identique entre tous ces objets, et en ce qu'il consiste à valoriser ces paramètres pour constituer un facteur commun dans ce modèle.

4- Procédé selon l'une des revendications 1 à 3, caractérisé en ce qu'il consiste, lors de l'écriture du fichier de configuration, si au moins deux objets se reposent sur le même modèle, à utiliser le facteur commun et à valoriser uniquement le restant des paramètres de ce modèle autant de fois qu'il y a d'objets se reposant sur ce modèle d'élément.

5- Procédé selon l'une des revendications 1 à 4, caractérisé en ce que le langage est extensible et en ce qu'il consiste à donner un nom pour identifier le modèle dans le fichier de description, et en ce qu'il consiste à inclure dans le modèle une référence du fichier de description, cette référence définissant les contraintes à respecter sur la structure et la syntaxe de ce modèle.

6- Procédé selon l'une des revendications 1 à 5, caractérisé en ce que le langage est extensible et en ce qu'il consiste à introduire dans un modèle deux mots clés DEFINIR et DEFINI indiquant qu'un paramètre est à définir (DEFINIR) ou est défini (DEFINI) dans ce modèle.

7- Procédé selon l'une des revendications 1 à 6, caractérisé en ce que le langage est le langage XML, et en ce qu'il consiste à prendre comme paramètre un élément et/ou un attribut d'un objet.

8- Procédé selon la revendication 7, caractérisé en ce qu'il consiste à étendre le fichier de description par au moins un modèle d'élément comprenant au moins un paramètre (élément et/ou attribut) décrit dans le fichier de description, et à valoriser tout ou partie des paramètres de ce modèle

9- Procédé selon la revendication 7 ou 8, caractérisé en ce qu'il consiste à donner un nom pour identifier le modèle d'élément dans le fichier de description, et en ce qu'il consiste à inclure dans le modèle une référence à un élément du fichier de description, cette référence définissant les contraintes à respecter sur la structure et la syntaxe de ce modèle.

10- Procédé selon l'une des revendication 7 à 9, caractérisé en ce qu'il consiste à inclure dans un modèle d'élément au moins un modèle d'élément.

11- Procédé selon l'une des revendications 7 à 10, caractérisé en ce qu'il consiste, sur requête d'une application utilisant le fichier de configuration, à transmettre le facteur commun et les blocs résultant de la valorisation des éléments indéfinis.

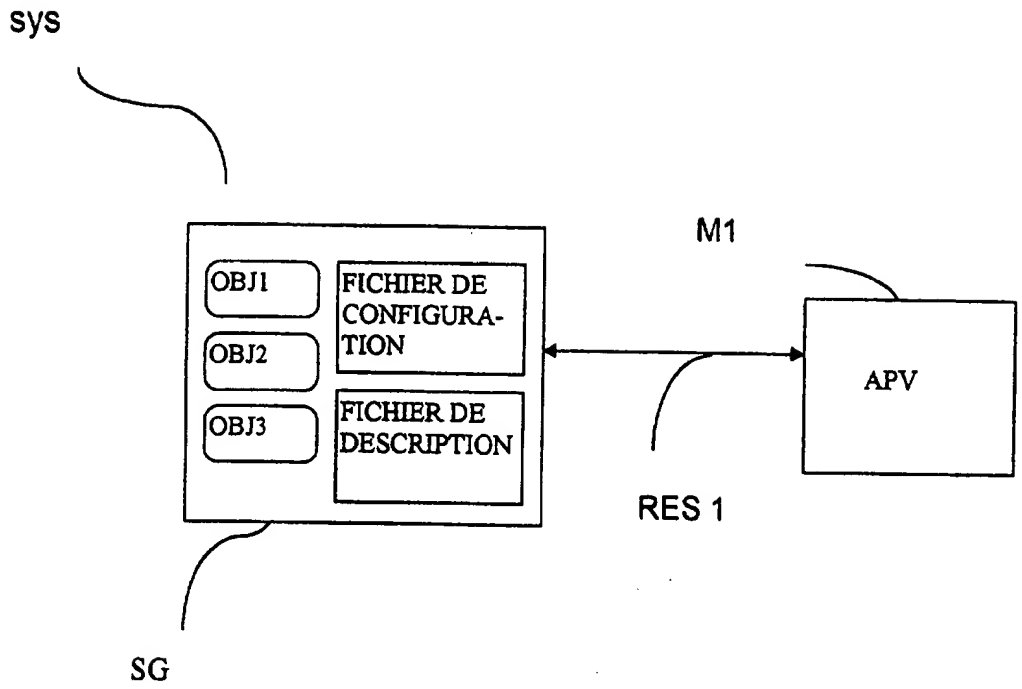


Figure 1

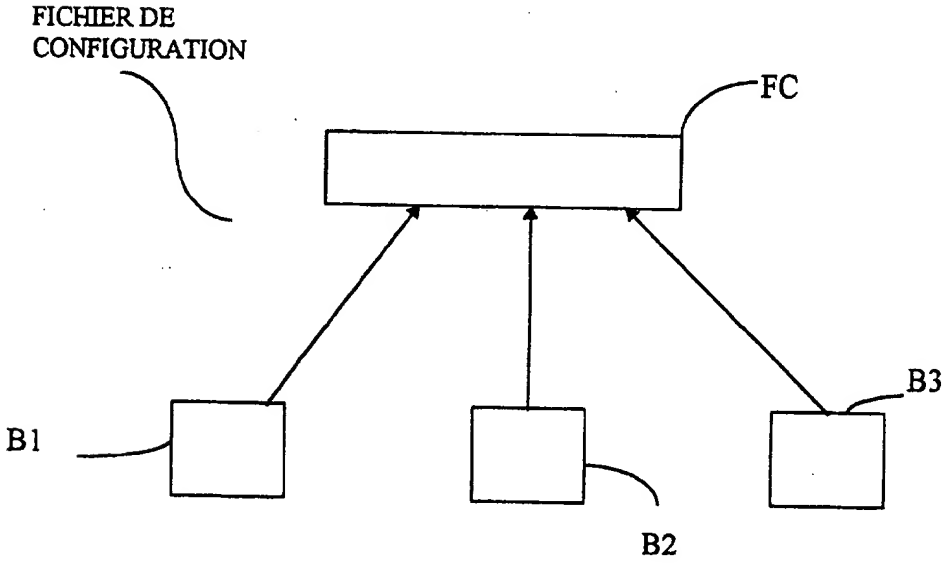


Figure 2

RÉPUBLIQUE FRANÇAISE

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

RAPPORT DE RECHERCHE PRÉLIMINAIRE

établi sur la base des dernières revendications déposées avant le commencement de la recherche

2801705

N° d'enregistrement national

FA 583176
FR 9914882

DOCUMENTS CONSIDÉRÉS COMME PERTINENTS		Revendication(s) concernée(s)	Classement attribué à l'invention par l'INPI
Catégorie	Citation du document avec indication, en cas de besoin, des parties pertinentes		
X	CLEMENS KERER: "A flexible and extensible security framework for Java code" INTERNET DOCUMENT: DIPLOMARBEIT, 'en ligne! 22 octobre 1999 (1999-10-22), XP002151129 Vienna, Austria. Extrait de l'Internet: <URL:http://www.infosys.tuwien.ac.at/Teaching/Finished/MastersTheses/JSEF/thesis.ps.zip> 'extrait le 2000-10-25!	1,4-7,9, 10	G06F19/00
A	* page 46, ligne 1 - page 47, ligne 22 * * page 85, ligne 1 - page 89, dernière ligne * * page 91, ligne 3 - page 95, dernière ligne *	2,3,8,11	

A	BERT BOS: "XML in 10 points" INTERNET DOCUMENT, 'en ligne! 27 mars 1999 (1999-03-27), XP002151130 Extrait de l'Internet: <URL:http://www.w3.org/XML/1999/XML-in-10-points> 'extrait le 2000-10-24! * page 1, ligne 10 - ligne 11 * * page 1, ligne 13 - ligne 17 * * page 2, ligne 12 - ligne 13 *	1-11	

A	ÉRIC JACOBONI (JACO@MAIL.DOTCOM.FR): "Lire et envoyer du courrier off-line sur sa machine" INTERNET DOCUMENT, 'en ligne! 1 juillet 1998 (1998-07-01), XP002151131 Extrait de l'Internet: <URL:ftp://anonymous:anonymous@ftp.linux-france.org/pub/article/mail/sendmail/sendmail.ps.gz> 'extrait le 2000-10-24! * page 2, ligne 18 - ligne 37 * * page 5, ligne 40 - page 8, ligne 30 *	1	

DOMAINES TECHNIQUES RECHERCHÉS (Int.CL.7)

G06F

Date d'achèvement de la recherche	Examinateur
26 octobre 2000	Ecolivet, S.

CATÉGORIE DES DOCUMENTS CITÉS

X : particulièrement pertinent à lui seul
Y : particulièrement pertinent en combinaison avec un autre document de la même catégorie
A : arrière-plan technologique
O : divulgation non-écrite
P : document intercalaire

T : théorie ou principe à la base de l'invention
E : document de brevet bénéficiant d'une date antérieure à la date de dépôt et qui n'a été publié qu'à cette date de dépôt ou qu'à une date postérieure.
D : cité dans la demande
L : cité pour d'autres raisons

& : membre de la même famille, document correspondant

EPO FORM 1503 12.99 (P04C14)